TUX 2.1

Reference Manual

Red Hat, Inc.

TUX 2.1: Reference Manual

TUX(EN)-2.1-RHI (2001-12-02T23:45-0400)

Table of Contents

Chapter 1. What is TUX?

TUX is a kernel-based web server licensed under the GNU General Public License (GPL).

It is currently limited to serving static web pages and coordinating with kernel-space modules, user-space modules, and regular user-space web server daemons to provide dynamic content. Regular user-space web servers do not need to be altered in any way for TUX to coordinate with them. However, user-space code has to use a new interface based on the tux(2) system call.

Although dynamic content is becoming increasingly popular, there is still a need to serve static content. For example, nearly all images are static. TUX can serve static content very efficiently from within the Linux kernel. A similar operation is already performed by the Network File System (NFS) daemon that runs in the kernel.

TUX also has the ability to cache dynamic content. TUX modules (which can be build in kernel space or in user space; user space is recommended) can create "objects" which are stored using the page cache. To respond to a request for dynamic data, a TUX module can send a mix of dynamically-generated data and cached pre-generated objects, taking maximal advantage of TUX's zero-copy architecture.

This new architecture for serving dynamic content requires a new API. The current API's for CGI can not be sufficiently mapped to TUX's API. Thus, existing CGI applications must be converted before TUX will process them. If the CGI application does not require the increased speed of TUX, TUX can process it by running the CGI application normally. This is done through TUX's CGI module. TUX can also handle a complex request (CGI or otherwise) by redirecting it to another web server daemon such as Apache. In other words, static content, TUX modules, old-style CGI applications, and programs specifically written for other webservers can be run on the same system with TUX as the main web server.

In summary, the differences between TUX and other webservers as well as the benefits of using TUX include:

- TUX runs partly within a custom version of kernel 2.4.x or higher and partly as a user-space daemon.
- With a capable network card, TUX enables direct scatter-gather DMA from the page cache directly to the network, thus avoiding data copies.
- Whenever TUX is unsure how to process a request or receives a request it is unable to handle, it always redirects the request to the user-space web server daemon to handle it in an RFC-compliant manner. An example of this user-space web server daemon is Apache.


Apache is used throughout this document as the user-space web server daemon for readability.

For questions or comments about TUX or this documentation, join the <tux-list@redhat.com> mailing list. For instructions on joining the mailing list, see http://www.redhat.com/mailing-lists/.

Also visit the Red Hat TUX Web Server Support page http://www.redhat.com/services/techsupport/application/tux.html.

1.1. New TUX 2.1 Features

The TUX 2.1 release is an incremental upgrade to TUX 1.0 and keeps source-code level compatibility with user-space modules.

The incremental enhancements include

- True zero-copy disk reads — Whereas TUX 1.0 copied files into a temporary buffer, TUX 2.1 is integrated with the page cache and thus uses zero-copy block IO.
- Generic zero-copy network writes — TUX 2.1 uses the generic zero-copy TCP framework.
- Zero-copy parsing — Where possible, TUX parses input packets directly. Even in RAM-limited situations, TUX now does full, back-to-back zero-copy I/O.

Other changes include

- Enhanced user-space utilities and module support.
- Mass virtual hosting support — The host-based virtual server patch has been added to TUX. There is no limit on the number of virtual hosts supported, only RAM and diskspace.
- CGIs can be bound to particular CPUs or can be left unbound.
- A number of bugs were fixed which caused performance problems — TUX 2.1 is now significantly faster than TUX 1.0!

1.2. Summary of System Requirements

- TUX Customized 2.4.x-based version of the kernel or higher
- x86, Alpha, IA64 or PowerPC/64 platform (should work on PowerPC/32, untested on Sparc)
- Alternate web server such as Apache running on the same server to process unknown requests

1.2.1. Current Limitations

- TUX can only call the other Web server such as Apache on the same server. In future revisions, it will allow the rollover of unsupported content to an alternate server.

Chapter 2. Installation

This chapter describes how to install TUX.

2.1. Installation Instructions

1. For optimal performance, create a separate RAID partition as the document root for TUX.
2. Configure and install the kernel with TUX support built-in, if it has not already been provided with TUX configured.
3. Install the TUX package with the command `rpm -Uvh tux-2.1.0-2.i386.rpm` (modify as necessary for new versions...)
4. Create an `index.html` file in `/var/www/html`, the default document root directory.
5. Start TUX with the command `service tux start` (or `./tux.init start` on Linux systems not running Red Hat Linux), and test the URL `http://localhost/` with **lynx** or any Web browser.

The latest TUX releases can be downloaded from http://people.redhat.com/~mingo/TUX-patches/. To install a TUX patch use the following instructions:

1. Download the latest TUX patch and userspace utilities from http://people.redhat.com/~mingo/TUX-patches/.
2. Apply the TUX patch to a vanilla 2.4.2 kernel tree with the command `patch -p0 < tux2-full-2.4.2-X6` (where `tux2-full-2.4.2-X6` is the TUX kernel patch).
3. Use `make oldconfig` to enable TUX in the kernel config, compile it, and boot into the TUX kernel.
4. Compile and install the userspace utilities, where `tux-2.0.25` is the version of TUX you want to install:

```
tar xzvf tux-2.0.25.tar.gz
cd tux-2.0.25
make
make install
```

5. Create an `index.html` file in `/var/www/html`, the default document root directory.
6. Start TUX with the command `service tux start` (or `./tux.init start` on Linux systems not running Red Hat Linux), and test the URL `http://localhost/` with **lynx** or any Web browser.

2.1.1. Log Files

For each request, TUX logs the address of the requestor, a date and time stamp accurate to at least one second, specification of the file requested, size of the file transferred, and the final status of the request.

The log files for TUX are stored in `/var/log/tux` in binary format. In this binary format, the log files are approximately 50% smaller than standard ASCII text log files. To view log files use the command `/usr/sbin/tux2w3c /var/log/tux`. The `tux2w3c` program converts the

binary log files into into standard W3C-conforming HTTPD log files. If you want to save the ASCII output, you can redirect the output to a file: `/usr/sbin/tux2w3c /var/log/tux > tux.log`, where `tux.log` is the name of the output file.

Sample log file output:

```
195.4.12.3 - - Fri Nov  9 01:05:56 2001 "GET /test.html HTTP/1.1" - 53 200
195.4.12.3 - - Fri Nov  9 01:06:10 2001 "GET / HTTP/1.1" - 2890 200
255.255.255.255 - - Fri Nov  9 01:06:10 2001 "GET /icons/apache_pb.gif HTTP/1.1" -
0 404
195.4.12.3 - - Fri Nov  9 01:06:10 2001 "GET /poweredby.png HTTP/1.1" - 1154 200
195.4.12.3 - - Fri Nov  9 01:06:04 2001 "GET /test.html HTTP/1.1" - 53 200
195.4.12.3 - - Fri Nov  9 01:06:22 2001 "GET /manual/index.html HTTP/1.1" - 5557 200
195.4.12.3 - - Fri Nov  9 01:06:04 2001 "GET /test.html HTTP/1.1" - 53 200
195.4.12.3 - - Fri Nov  9 01:06:22 2001 "GET /manual/images/apache_header.gif HTTP/1.1"
4084 200
195.4.12.3 - - Fri Nov  9 01:06:04 2001 "GET /test.html HTTP/1.1" - 53 200
195.4.12.3 - - Fri Nov  9 01:06:22 2001 "GET /manual/images/pixel.gif HTTP/1.1" -
61 200
195.4.12.3 - - Fri Nov  9 01:06:04 2001 "GET /test.html HTTP/1.1" - 53 200
195.4.12.3 - - Fri Nov  9 01:06:26 2001 "GET /manual/invoking.html HTTP/1.1" -
1 200
195.4.12.3 - - Fri Nov  9 01:06:04 2001 "GET /test.html HTTP/1.1" - 53 200
195.4.12.3 - - Fri Nov  9 01:06:35 2001 "GET /manual/stopping.html HTTP/1.1" -
1 200
195.4.12.3 - - Fri Nov  9 01:06:04 2001 "GET /test.html HTTP/1.1" - 53 200
195.4.12.3 - - Fri Nov  9 01:06:37 2001 "GET /manual/howto/ssi.html HTTP/1.1" -
18523 200
195.4.12.3 - - Fri Nov  9 01:06:41 2001 "GET /manual/new_features_1_3.html HTTP/1.1" -
34531 200
```

Chapter 3. Configuration

This chapter describes how to configure the TUX Web Server.

3.1. Modes of Operation

The recommended mode of operation is to have TUX running as the main web server and Apache run as the assistant.

- Client Port: 8080 (or other)
- Web Server Port: 80

For the recommend mode where TUX is the main web server, the configuration for the user-space daemon must be changed to use port 8080. For Apache configuration, the changes are made in the configuration file `/etc/httpd/conf/httpd.conf` by changing the line

```
Port 80
```

to

```
Port 8080
```

For security reasons, the line

```
BindAddress *
```

should be changed to

```
BindAddress 127.0.0.1
```

This will prevent outside users from accessing Apache directly. You must restart Apache for the changes to take effect with the command `/etc/rc.d/init.d/httpd restart`.

The alternate mode of operation is to have the user-space daemon such as Apache as the main web server and TUX as the assistant.

- Client Port: 80
- Web Server Port: 8080 (or other)

3.2. Compressed Gzip Data Stream

TUX is now able to send compressed (gzip) data. This has the potential to decrease the amount of data the Web server sends to the client browser and decrease the browser's load time.

By default, this data compression is disabled. To enable it, add the following line to `/etc/sysctl.conf`:

```
net.tux.compression=1
```

The Gzip file with the extension .gz must be in the same directory as the uncompressed versions of the pages you wish to serve. All of the following conditions must be true for TUX to send the .gz file. Otherwise, the original file(s) are sent.

- The TUX compression feature is on in `/etc/sysctl.conf`.
- The client has explicitly stated to support gzip encoding.
- The original file exists, is a regular file, and has the proper permissions.
- The .gz file exists, is a regular file, and has the proper permissions.
- The .gz file is newer than or has the same-date as the original file.
- The size of the .gz file is smaller than original file.


A cron job can be created to generate a new gzip file from the latest uncompressed data in each directory.

3.3. Parameters

This chapter describes how to configure TUX via the available TUX parameters.


Most parameters can only be set when TUX is not active.

3.3.1. /proc/sys/net/tux Parameters

The following parameters are set through `/proc/sys/net/tux`. Note this has changed from the original location of `/proc/sys/net/http` and `/proc/net/http`.

Table 3-1. TUX Configuration Parameters

Name	Default	Description
serverport	80	No longer available. To change the TUX HTTP server port, use the command echo `'http://0.0.0.0:80' > /proc/net/tux/0/listen/0`, where 80 is the port number.
clientport	8080	The port listened to by the userspace http-daemon
documentroot	/var/www/html	The directory where the web pages are stored. If using the init script `/etc/rc.d/init.d/tux`, documentroot should be set in `/etc/sysconfig/tux` as DOCROOT.

Name	Default	Description
http_subdocroot	No value set by default	The directory, relative to the documentroot, where the web pages are stored. TUX defaults to using documentroot if http_subdocroot has no value.
ftp_subdocroot	No value set by default	The directory, relative to the documentroot, where the files to be served by the FTP server are stored. TUX defaults to using the document root defined for the HTTP server if ftp_subdocroot has no value.
ftp_log_retr_only	0	If set to 0, TUX will log every other command as well. If set to 1, TUX will only log RETR FTP commands to cut down the log size.
ftp_wait_close	1	If set to 1, TUX will wait for data socket to close before sending completion message to command socket. Certain clients (for example, **lynx**) get confused by TUX's high level of asynchronity. This setting slows down FTP RETR downloads and directory listings and increases packet count, but it works around broken FTP clients. If set to 0, TUX will not wait for the FTP client to notice the closed data socket.
404_page	404.html	If TUX does not manage to look up a requested page then it first tries to look up the document specified in 404_page. If the 404 page can not be found, the canned 404 message is sent. The file is relative to the document root.
threads	The number or server-threads, set at most to 1 per CPU	The number of kernel threads (and associated daemon threads) to be used. Can not be greater than the number of CPUs on the system. If using the init script `/etc/rc.d/init.d/tux`, threads should be set in `/etc/sysconfig/tux` as TUXTHREADS.
mode_allowed	S_IROTH	Required permissions for files TUX will process. See "man 2 stat" for all values.
mode_forbidden	dir+sticky+execute	Files with this permission-mask are "forbidden" and will not be processed by TUX. See "man 2 stat" for all values.
nonagle	2	If set to 0, standard Nagle output packet merging. If set to 1, no Nagle merging of output packets. If set to 2, TCP_CORK-style output packet merging.

Name	Default	Description
push_all	0	If set to 0, may merge subsequent packets. If set to 1, force a packet boundary right after the end of the TUX request.
compression	0	If set to 0, it is disabled. If set to 1, sending gzip compressed data is turned on. See Section 3.2 for details.
cgi_uid	-1	UID as which to run CGI programs. Set by default to the ID for "nobody" in the tux init script.
cgi_gid	-1	GID as which to run CGI programs. Set by default to the ID for "nobody" in the tux init script.
cgiroot	/var/www/tux/cgiroot	The directory in which TUX runs CGI programs. Set by default to $DOCROOT in the tux init script.
cgi_cpu_mask	0xffffffff	The default value allows CGI scripts to execute on all CPUs. This value can be set to bind newly started CGI scripts to a single CPU or a set of CPUs. The CPUs are represented in a 32-bit bitmask, where bit 1 is CPU#0, bit 2 is CPU#1, etc. This value has not effect on single-processor systems.
cgi_inherit_cpu	0	If set to 1, all newly started CGI scripts inherit the CPU-binding of the CGI-starting TUX thread — all processes started by the CGI script will be bound to the same CPU as the parent CGI.
max_connect	1000	Maximum number of concurrent connections.
max_header_len	3000	Maximum header size in bytes.
max_output_bandwidth	0	Maximum output bandwidth (per connection) used up by keepalive requests in bytes/sec. The default value of 0 means off or unlimited bandwidth. Can be as low as 1 byte/sec. This parameter replaces max_keepalive_bw.
max_keepalive	1000	Maximum number of open keepalive connections. After having reached max_keepalives connections, TUX zaps old connections based on LRU.

Name	Default	Description
keepalive_timeout	0	Unfinished and should not be used. A per-client-connection timer that will time out if a request does not arrive within a pre-specified time. Timeout value is set in seconds.
max_object_size	100MB	Maximum file size TUX is willing to serve.
Dprintk	0	If TUX_DEBUG is turned on, then print out very verbose messages to syslog. Should only be used for debugging purposes.
ack_pingpong	1	Delay TCP ACK for incoming frames in the hopes of a subsequent output frame. Separate ACK will happen nevertheless, if no output frame is generated within a timeout.
all_userspace	0	If set to 1, every complete and valid HTTP request will be bounced to the first user-space module. The user-space module "takes control" over the entire URL space. Then, the user-space module can make a decision to 1) serve a static reply, 2) serve a cached dynamic reply, or 3) create a dynamic reply. If set to 0, all_userspace is disabled.
application_protocol	0	If set to 1, it enables the TUX FTP server. If set to 0, this feature is disabled. Refer to Section 3.9 for details.
logentry_align_order	N/A	Currently unused.
logfile	/var/log/tux	The filename of the TUX binary logfile. Refer to Section 2.1.1 for more information.
logging	0	If set to 1, logging is enabled. If set to 0, logging is disabled.
redirect_logging	1	Set to 0 to suppress redirected connections. Can be changed at runtime and takes effect immediately.
referer_logging	0	If set to 1, referer logging is enabled and will be automatically printed by `tux2w3c` if the referer entry is present. If set to 0, referer logging is disabled.
max_backlog	2048	Maximum size of SYN backlog of the TUX listening socket.

Name	Default	Description
virtual_server	0 (off)	Turns on mass virtual hosting. Hosts are headers from the browser that are directly turned into $DOCROOT/<Host> 'virtual docroots.' This way any number of hosts can be served by a single TUX server without any performance penalty at all. Refer to Section 3.8 for details.
mass_hosting_hash	0 (off)	If virtual_server is enabled, this parameter modifies the hostname mapping to be more effective for a large number of hosts. Refer to Section 3.8 for details.
strip_host_tail	0 (off)	If virtual_server is enabled, this parameter strips off hostname components. Refer to Section 3.8 for details.
zerocopy_parse	1	Use the input packet buffer as a temporary buffer and avoids copying input data.
defer_accept	0 (disabled if keepalive_timeout or max_keepalives is set)	If set to 1, then TUX processes will not be woken up on the initial SYN-ACK event of a new TCP connection, but only after the first real data packet has arrived. If set to 0, this feature is disabled.
http_dir_indexing	0 (disabled)	If set to 1, TUX will list files in readable directories if an index file does not exist.

3.3.2. Init Script Parameters

If the TUX init script `/etc/rc.d/init.d/tux` is used, the following parameters can be set in the file `/etc/sysconfig/tux` (see Table 3-2). They should *not* be set in `/etc/sysctl.conf` because the init script will override parameters set in `/etc/sysctl.conf`. Using the init script is the preferred method for starting TUX.

Table 3-2. `/etc/sysconfig/tux` parameters

Parameter	Default	Description
TUXTHREADS	The number of server-threads, set at most to 1 per CPU	The number of kernel threads (and associated daemon threads) to be used, cannot be greater than the number of CPUs on the system

Parameter	Default	Description
DOCROOT	/var/www/html	The document root, the directory where the web pages are stored.
CGI_UID	nobody	UID (user) as which to run CGI programs.
CGI_GID	nobody	GID (group) as which to run CGI programs.
DAEMON_UID	nobody	UID (user) as which the daemon runs.
DAEMON_GID	nobody	GID (group) as which the daemon runs.
CGIROOT	/var/www/html	The directory where the CGI programs are stored. CGI programs can be started in the chroot environment by default. Set CGIROOT=/ if you want CGI programs to have access to the whole system.
MAX_KEEPALIVE_TIMEOUT	30	Timeout value for each HTTP connection. Use this to prevent connection hangs.
TUXMODULES	demo.tux demo2.tux demo3.tux demo4.tux	list of user-space loadable TUX modules, see `man 2 tux` for more information
MODULEPATH	/	Path to the user-space loadable TUX modules

3.3.3. /proc/net/tux Parameters

After starting TUX, the `/proc/net/tux` directory contains the file `stat`. This file contains statistics on every allocated request structure. As this works even if `TUX_DEBUG` is turned off, this should help debugging things a bit more. It can also be used to calculate file download status. For example, TUX/FTP - the 100*f_pos/filelen gives the current progress of download.

It is possible to bind the logger thread to any particular CPU (or group of CPUs), so you can localize IO, via `/proc/net/tux/log_cpu_mask` The default is to run on any CPU.

3.3.4. Required Parameters

Before starting TUX, the following parameters must be set:

- serverport
- clientport
- DOCROOT

The DOCROOT for TUX must be the same document root directory as Apache or other user-space daemon running as the assistant web server for TUX to properly redirect requests.

3.4. Starting TUX

TUX can be started by issuing the command `/etc/rc.d/init.d/tux start`.

This script is written to start TUX on a single-processor as well as a multi-processor server.

If you choose to write your own script to start TUX or start it from the `/usr/sbin/tux` binary, you can use the following options:

Table 3-3. /usr/sbin/tux options

Option	Description
-t, --threads=*N*	number of tux threads
-d, --docroot=*path*	directory path for document root
-m, --modpath=*path*	directory path for user-space loadable TUX modules
-d, --daemon	run in the background as a daemon
-D, --date-interval=*seconds*	how often (in seconds) to update the date string, the default is 1 second
-?, --help	show help message
--usage	display brief usage message

IRQ affinity is a small performance boost. If you are not experiencing any performance difficulties, it is not recommended you try the following.

3.4.1. IRQ Affinity

Binding IRQ's to a group of CPU's is a new feature of the 2.4 kernel. While it was originally developed as part of TUX, it is now a generic and independent kernel feature. Every IRQ source in Linux has an entry in `/proc/irq` directory. For example, the settings for IRQ 40 is stored in `/proc/irq/40`. IRQ affinity, or IRQ bindings, is configured though the smp_affinity setting in that directory. For example, the smp_affinity for IRQ 40 is in `/proc/irq/40/smp_affinity`. The value of the smp_affinity setting is a bitmask of all CPU's that are permitted as a resource for the given IRQ. The default value for smp_affinity is the HEX value `0xffffffff`. This means the processes for the IRQ are sent to all CPU's. You are not allowed to turn off all CPU's for an IRQ. If the IRQ controller does not support IRQ affinity, the value can not be changed from the default. If multiple CPU's are defined, then the IRQ source uses the least busy CPU. This is called 'lowest priority APIC routing.' IRQ affinity is achieved by binding an IRQ to a specific CPU or group of CPU's by echoing a HEX value to smp_affinity for the IRQ.

Thus, TUX thread N is bound to CPU N. If a single TUX thread is used (which is recommended) and there is only one network interface card, then the network interface card's IRQ should be bound to CPU0.

3.5. Stopping TUX

If TUX was started with the `/etc/rc.d/init.d/tux start` script, stop TUX by executing the `/etc/rc.d/init.d/tux stop` script. This will unload all user-space TUX modules automatically.

If you did not use the scripts provided, stop TUX with the command `/usr/sbin/tux -s` or `/usr/sbin/tux --stop`.

3.6. Debugging TUX

To print out the state and various other information about TUX, execute the `gettuxconfig` script. You must be root to run this script.

The `checkbindings` shell script checks an existing TUX SMP configuration, whether all IRQ, interface, and listening socket bindings and affinities are set up correctly. It assumes that the interfaces eth0, eth1, eth2, and so on are used linearly and mapped linearly. The script warns if it finds any inefficiency.

3.7. MIME Types

TUX supports three types of MIME types starting with version 2.0.13 and kernel patch 2.4.2-P3. They are defined in `/etc/tux.mime.types`.

Table 3-4. MIME Types

MIME Type	File Extension	Description
TUX/redirect	`pl` `php`	All extensions listed after TUX/redirect will be redirected to the secondary server.
TUX/CGI	`cgi` `pl`	All extensions listed after TUX/CGI will be handled by the TUX CGI engine directly.
TUX/module	`tux` `x`	All extensions listed after TUX/module will be handled by TUX userspace modules.

The TUX/redirect MIME type will redirect all requests to files ending in `.pl` or `.php` to Apache, without having to check for file permissions.

The `TUX/CGI` MIME type specify scripts that should be located in the `$DOCROOT/cgi-bin` directory, or the directory specified by the `cgiroot` parameter. Refer to Section 3.3 for details.

Refer to Chapter 5 for details about the `TUX/module` MIME type.

3.8. Mass Virtual Hosting

TUX supports mass virtual hosting, which enables a very high number of virtual hosts.

There are three tunables that deal with virtual hosting:

- `virtual_server` — Valid values are 0, 1, 2, or 3.
- `mass_hosting_hash` — Valid values are 0, 1, 2, or 3.
- `strip_host_tail` — Value must be an integer.

These three tunables depend on each other, and the `strip_host_tail` tunable is only used if host based virtual serving is enabled. Otherwise, it is ignored.

3.8.1. `virtual_server`

If the value is set to 0, virtual hosting is disabled:

```
http://www.example.com/a.html => $DOCROOT/a.html
```

If the value is set to 1, host-based virtual hosting is enabled:

```
http://www.some.site.com/a.html => $DOCROOT/some.site.com/a.html
```

Note

TUX strips off the www. prefix variants and transforms the hostname to lowercase.

If the value is set to 2, IP-based virtual hosting is enabled:

```
http://www.some.site.com/a.html => $DOCROOT/1.2.3.4/a.html
```

If the value is set to 3, a mixture of host-based and IP-based virtual hosting is enabled:

```
http://www.some.site.com/a.html => docroot/1.2.3.4/some.site.com/a.html
```

3.8.2. `mass_hosting_hash`

The `mass_hosting_hash` tunable modifies the hostname mapping to be more effective for a large number of hosts.

If the value is set to 0, `mass_hosting_hash` is disabled.

If the value is set to 1:

```
http://www.some.site.com/a.html => docroot/s/some.site.com/a.html
```

If the value is set to 2:

```
http://www.some.site.com/a.html => docroot/s/so/some.site.com/a.html
```

If the value is set to 3:

```
http://www.some.site.com/a.html => docroot/s/so/som/some.site.com/a.html
```

3.8.3. `string_host_tail`

The `strip_host_tail` tunable strips off hostname components, starting at the end of the hostname.

If the value is set to 0, this tunable is disabled.

If the value is set to 1:

```
http://www.some.site.com/a.html => docroot/some.site/a.html
```

If the value is set to 2:

```
http://www.some.site.com/a.html => docroot/site/a.html
```

and so on...

3.9. TUX as an FTP Server

Starting with version 2.0.21 of TUX and version 2.4.2-U7 of the TUX patched kernel, TUX can be configured to run as an anonymous FTP server.

To use TUX as an HTTP and FTP server at the same time, use the following commands:

```
echo "http://0.0.0.0:80" > /proc/net/tux/0/listen/0
echo "ftp://0.0.0.0:21" > /proc/net/tux/0/listen/1
```

By default, the document root for the FTP server is the document root for the HTTP server set as `DOCROOT` in `/etc/sysconfig/tux` or the value of `/proc/sys/net/tux/documentroot`.

To configure different document roots for the HTTP and FTP server, set the DOCROOT in `/etc/sysconfig/tux` and execute the following commands:

Note

The http_subdocroot and ftp_subdocroot are relative to DOCROOT.

```
echo '/www/' > /proc/sys/net/tux/http_subdocroot
echo '/ftproot/' > /proc/sys/net/tux/ftp_subdocroot
```

Restart TUX to apply the changes:

```
service tux restart
```

After executing these commands, the TUX FTP server will be running on port 21.

To have it display directory listings, run the `generatetuxlist` script from the FTP docroot. This script creates the files `.TUX-LIST` and `.TUX-NLIST` files that cache the directory listing. Everytime the FTP docroot directory changes, the script must be re-run to generate an updated directory listing.

Note

The TUX FTP server has been through numerous stresstests and FTP-client compatibility tests. However, it is still early software. TUX/FTP has no known bugs or security holes at the moment. It has not been tested with a wide number of FTP clients yet (only the most obvious ones).

3.9.1. TUX FTP Server Security Features

The following are security features of the TUX FTP Server:

- Because TUX does not start per-client processes, the memory allocation overhead for each FTP client logged in is less than 10 KB. This allows thousands of parallel connections.
- Paranoid parser and paranoid command-evaluation.
- Chroots to docroot.
- Never starts any external userspace process. All FTP functionality is done in a approximately 900 lines C module, in the kernel.
- Even in kernel mode the TUX FTP Server drops all priviledges and switches to uid and group `nobody`.
- Only the most trivial globbing (`mget *`) supported, and no recursion support.

Chapter 4. Security

TUX is designed to have very strict security. This is possible because the assistant user-space daemons is used to handle the complex exceptions.

TUX only serves a file if

1. The URL does not contain `?`.
2. The URL does not start with `/`.
3. The URL points to a file that exists.
4. The file is world-readable. [1]
5. The file is not a directory. [1]
6. The file is not executable. [1]
7. The file does not have the sticky-bit set. [1]
8. The URL does not contain any forbidden substrings such as `..` [1]

1. Configurable through the sysctl parameters in `/proc/sys/net/tux`
1. Configurable through the sysctl parameters in `/proc/sys/net/tux`
1. Configurable through the sysctl parameters in `/proc/sys/net/tux`
1. Configurable through the sysctl parameters in `/proc/sys/net/tux`
1. Configurable through the sysctl parameters in `/proc/sys/net/tux`

Chapter 5. User-space Loadable Modules

In addition to parts of TUX running in kernel-space, user-loadable modules can also be written for TUX.

The API for the user-loadable modules is currently under development. This section of the documentation will be updated as the API becomes available.

User-space loadable modules are usually a single `.c` file and are compiled as a shared libraries as a `.so` file. There can be an unlimited number of user-space HTTP modules, and they can be compiled in a language of choice. They have full address space protection, can not crash the kernel, and are unpriviledged.

A list of user-space loadable TUX modules and their location must be specified with the `TUXMODULES` parameter in `/etc/sysconfig/tux`. Refer to Section 3.3 for details.

Starting with TUX version 2.0.13 and kernel patch 2.4.2-P3, user-space loadable modules do not require special permissions to be activated. Instead, the module is specified using a common MIME type definition file. The user-space modules must end with the file extension `.tux` or `.x` and specified with the `TUXMODULES` parameter in `/etc/sysconfig/tux`. The file must be owned by *root* with *root* as the group and must be world-readable. It does not have to be executable. For example, to use the `demo.tux` user-space loadable module, you might have the following file:

```
[root@m /]# ls -l /var/www/html/demo.tux
-rw-rw-r--    1 root     root            0 Sep  3 04:42 /var/www/html/demo.tux
```

If TUX finds a URL object that has this MIME type, it searches the internal list of modules defined as TUXMODULES in `/etc/sysconfig/tux`. If there is a match, TUX sends the request to the user-space loadable module.

For further information about writing a TUX user-space loadable module, see the file `/usr/share/doc/tux-<version>/TUXAPI-user.txt`.

Index

Symbols

B

C

D

F

G

I

L

M

P

S

T

V

W